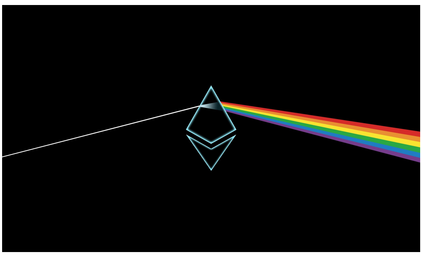
zarytskiy 21 октября в 17:16

Как работает Эфириум (Ethereum)?

Криптовалюты

Из песочницы



Введение

Наверняка некоторые из вас знают, что такое блокчейн Эфириум (с англ. Ethereum), другие, напротив, не имеют даже малейшего представления о нем. Так или иначе, и первые и вторые что-нибудь да слышали о данной платформе. В последнее время этой теме было посвящено множество статей в различных крупных журналах, однако для тех людей, кто мало что слышал об Эфириуме, все статьи на эту тему представляются чем-то мистическим и совершенно непонятным. Тогда, что же собой представляет данная платформа? Если вкратце: Эфириум – это общедоступная база данных с возможностью хранения цифровых транзакций в течение неограниченного времени. Важно также отметить, что для обслуживания и защиты такой базы данных не требуется каких-либо систем управления ключами. Вместо этого данная платформа работает как «беззащитная» транзакционная система – фреймворк, в котором физические лица могут совершать одноранговые транзакции, при этом ни одна из сторон не несет перед другой или третьей сторонами каких-либо обязательств.

Я не удивлюсь, если вы мало что поняли. Собственно, цель данной статьи – объяснить, каким образом блокчейн Эфириум функционирует на техническом уровне, не прибегая для этого к сложным математическим расчетам или ужасающим своей величиной формулам. Даже если вы и не программист, у меня есть полная уверенность в том, что данная статья поможет вам понять принципы технологии Эфириума. И даже если некоторые части данной статьи будут напичканы техническими определениями, которые могут показаться вам чересчур сложными для восприятия, вам не стоит отчаиваться, ведь ее цель – донести до вас понимание данной платформы в целом, не вдаваясь в технические и математические тонкости.

Многие из тем, затронутых в данной статье, представляют собой разжевывание тех основных понятий, с которыми вы, вероятно, уже не раз сталкивались, читая yellow paper (от англ. «желтая бумага» – официальная спецификация для Эфириума). Мною были добавлены собственные пояснения и диаграммы для того, чтобы вы как можно быстрей разобрались с данной технологией. Ну, а для самых храбрых и технически подкованных я могу посоветовать прочтения Ethereum yellow paper.

Давайте уже начнем!

Что такое блокчейн

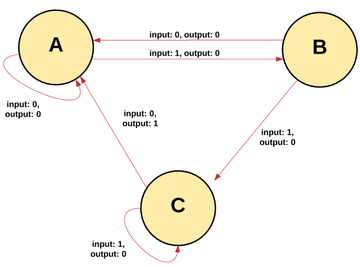
Блокчейн – это криптографически безопасная транзакционная одноэлементная система с общим состоянием. Далеко не самое простое определение, не так ли? Давайте разобьем каждую составляющую этого определения на отдельные части.

- «**Криптографически безопасный**» означает, что безопасность криптовалюты обеспечивается сложными математическими алгоритмами, которые практически невозможно обойти. Защита, выстроенная с помощью данных алгоритмов, представляет собой подобие файрвола: благодаря используемым алгоритмам обход системы безопасности практически невозможен (например, невозможно создание поддельных транзакций, уничтожение транзакций и т. д.).
- «**Транзакционная одноэлементная система**» означает, что существует только одно заданное состояние системы, благодаря которому происходят все транзакции, создаваемые в данной системе. Другими словами, для данной системы предусмотрено только одно состояние, которое является единственно верным.
- «**С общим состоянием**» означает, что состояние, заданное в системе, является общим и открытым для всех.

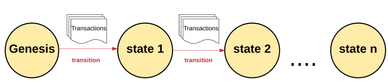
Таким образом, в платформе Эфириум реализуется приведенная выше парадигма блокчейна.

Парадигма блокчейна платформы Эфириум

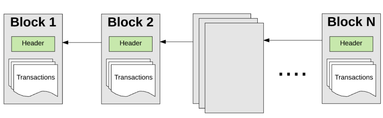
Блокчейн Эфириум является, по сути, **системой состояния транзакций**. В информатике такое понятие, как «система состояний» или «машина состояний» – это система, которая обрабатывает вводимую информацию и на основании последней преобразуется в новое состояние.



В машине состояний Эфириума все процессы начинаются с «первоначального состояния». Такое состояние представляет собой аналог нулевого состояния, в котором находится машина до того момента, как в ее сети начнут происходить какие-либо действия, связанные с транзакциями. Когда такие действия начнут происходить, первоначальное состояние заменяется на конечное, при этом в любой момент времени конечное состояние отображает текущее состояние Эфириума.



Состояние Эфириума имеет миллионы транзакций. Эти транзакции сгруппированы в «блоки». Блок содержит ряд транзакций, при этом каждый последующий блок соединен с предыдущим, благодаря чему обеспечивается своеобразная цепочка блоков.



Транзакция должна быть корректной, для того чтобы вызвать ее переход из одного состояния в другое. **Транзакция считается корректной только тогда, когда она прошла процесс проверки – так называемый «майнинг»**. Майнинг — это когда группа узлов (компьютеров) расходует свои вычислительные ресурсы для создания блока корректных транзакций.

Любой узел в сети, объявляющий себя майнером, может попытаться создать и проверить блок транзакций. Распространенным опытом является попытки множества майнеров одновременного создания и проверки блока транзакций. Каждый майнер предоставляет свое математическое «доказательство» при отправке блока в блокчейн, и это доказательство выступает в роли своеобразной гарантии: в случае если доказательство существует, транзакции в блоке считаются корректными.

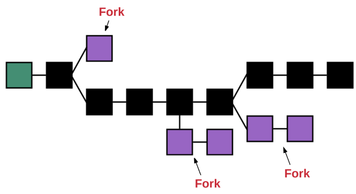
Майнер должен предоставить свое математическое доказательство быстрее, чем это сделает любой другой конкурент, для того чтобы его блок был добавлен в основной блокчейн. Процесс проверки каждого блока, который заключается в предоставлении майнером своего математического доказательства, называется «доказательством работы».

Майнер, который обосновывает новый блок, получает определенное вознаграждение за выполнение этой работы. О каком вознаграждении идет речь? В блокчейне Эфириума используется встроенный цифровой токен, который носит название «эфир» (от англ. ether— «эфир»). Каждый раз, когда майнер обосновывает свой блок транзакций, создается новый токен или новый эфир, а майнер получает вознаграждение за его создание.

Тогда, у вас может возникнуть вполне логичный вопрос: где гарантия того, что каждый майнер будет придерживается только одной цепочки блоков? Как мне убедиться в том, что другая команда майнеров не решит создать свою собственную цепочку блоков?

В самом начале данной статьи мы уже приводили такое понятие как «транзакционная одноэлементная система с общим состоянием». Исходя из такого определения можно сделать вывод, что не бывает двух и более корректных текущих состояний – оно является единственным в своем роде. Таким образом, каждый, кто принимает участие в процессе обоснования новых блоков, должен принять это утверждение за истину. Наличие нескольких состояний (или цепей) разрушило бы всю систему, потому что было бы невозможно договориться о том, какое из состояний является корректным. Например, представим, что существовало бы несколько цепочек блоков. Тогда, в теории, вы могли бы собрать 10 монет на одной цепочке, на другой – 20 монет, на третьей – 40 монет и т.д. В таком случае было бы невозможно определить, какая цепь является наиболее «корректной».

Всякий раз, когда генерируются несколько путей, возникает «разветвление». Зачастую, разветвления очень нежелательны, поскольку они нарушают целостность системы, а пользователям приходится выбирать одну из возможных цепочек.



Чтобы определить, какой из возможных путей является корректным, и предотвратить образования множества цепей, в Эфириуме применяется метод, называемый «**протокол GHOST**».

GHOST – «Жадное-и-Самое-Весомое-из-Известных-Дочерних-Деревьев» (Greedy Heaviest Observed Subtree)

Попробую объяснить простыми словами: **протокол GHOST объявляет, что мы должны выбрать только тот путь, на котором было выполнено наибольшее число вычислений**. Для определения такого пути можно использовать номер того блока, который был определен последним («листовой блок»). Благодаря такому подходу можно определить общее число блоков, находящихся в текущем пути (без учета блока первоначального состояния). Чем выше находится блок, тем длиннее путь и тем больше обоснований должны предоставить майнеры. Исходя из таких соображений, принимается единственно верная версия для текущего состояния.

Теперь, когда вы уже имеете представление о том, что такое блокчейн, я предлагаю разобраться с основными компонентами, из которых состоит система Ethereum:

- учетные записи
- состояние
- горючее о вознаграждение
- транзакции
- блоки
- выполнение транзакций
- майнинг
- обоснование

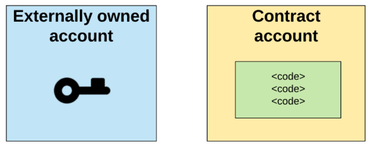
Небольшое отступление перед тем, как мы начнем: при упоминании хэша X имеется ввиду хэш KECCAK-256, используемый в Эфириуме.

Учетные записи

Глобальное общее состояние платформы Эфириум состоит из множества небольших объектов – учетных записей, которые взаимодействуют между собой за счет парадигмы обмена сообщениями. У каждой учетной записи есть определенное состояние и 20-байтовый адрес. Адресом в Эфириум является 160-битный идентификатор, используемый для идентификации любой из учетных записей.

Всего существует два вида учетных записей:

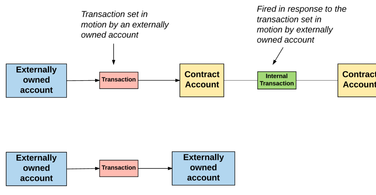
- Внешние учетные записи, контролируются с помощью закрытых ключей. При этом такие записи не имеют никакого кода, связанного с ними.
- Контрактные учетные записи, контролируются специальным кодом, указанным в условиях контракта, и имеющие связанный с ними код.



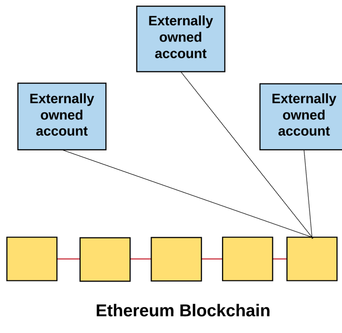
Внешние и контрактные учетные записи

Давайте разберемся с основными отличиями между внешними и контрактными учетными записями. **Для внешней учетной записи предусмотрена возможность отправлять сообщения другим внешним учетным записям, а также другим контрактным учетным записям**. Для данной цели необходимо создать и зарегистрировать новую транзакцию, используя закрытый ключ. Сообщение между двумя внешними учетными записями является всего лишь значением для передачи. С другой стороны, сообщение, отправленное от внешней учетной записи к контрактной, подразумевает активацию кода контрактной учетной записи, при этом появляется возможность совершения определенных действий (например, с помощью такого сообщения можно переводить токены, записывать значения во встроенную память, создавать токены, выполнять некоторые вычисления, создавать новые контракты и т. д.).

С помощью контрактных учетных записей, в отличие от внешних, самостоятельно инициировать новые транзакции невозможно. Вместо этого с помощью контрактных учетных записей можно только запускать транзакции в ответ на другие полученные транзакции (например, полученные из внешней учетной записи или из другой контрактной учетной записи). Более подробно о вызовах между контрактными учетными записями мы остановимся в разделе «Транзакции и сообщения».



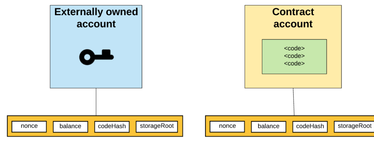
Каждое действие в блокчейне Эфириума происходит благодаря транзакциям, инициируемым внешне контролируемыми учетными записями.



Состояние учетных записей

Состояние каждой из учетных записей, вне зависимости от их типа, может принимать одно из четырех значений:

- **nonce**: Если настоящая учетная запись соответствует внешней учетной записи, то полученное число представляет собой количество транзакций, которые были отправлены с адреса учетной записи. Если учетная запись является контрактной учетной записью, то элемент nonce – это количество контрактов, созданных в данной учетной записи.
- **balance**: общее количество wei, приобретенных данной учетной записью. Например, каждый эфир, который является обменной единице Эфириума, содержит 10^18 wei – дробных частей эфира.
- **storageRoot**: хэш корневого узла префиксного дерева Меркла (что собой представляет дерево Меркла мы рассмотрим немного позже). Дерево Меркла кодирует хэш содержимого данной учетной записи, при этом по умолчанию оно является пустым.
- **codeHash**: хэш EVM-кода (от англ.Ethereum Virtual Machine; что это такое я расскажу немного позже) учетной записи. Для контрактных учетных записей данное поле является кодом, который хэшируется и хранится в виде codeHash.



Общее состояние системы

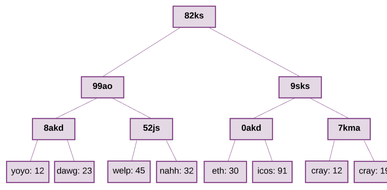
Итак, мы разобрались, что глобальное состояние Эфириума – это сопоставление адресам учетной записи состояний счета. Это сопоставление хранится в структуре данных – **префиксного дерева Меркла**.

Дерево Меркла (или «Merkle trie») представляет собой тип двоичного файла, состоящего из набора узлов, которые включают:

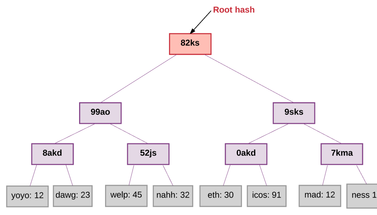
- определенное количество листовых узлов, которые располагаются в нижней части дерева,

содержащего базовые данные;

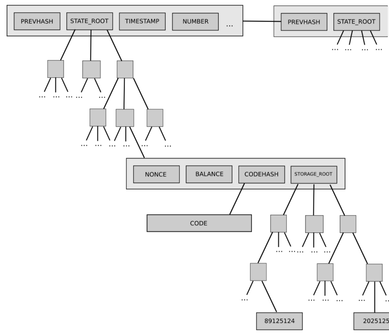
- набор промежуточных узлов, при этом каждый узел представляет собой хэш двух его дочерних узлов
- один корневой узел, также образованный из хэша двух дочерних узлов, который представляет вершину дерева



Данные, находящиеся в нижней части дерева, создаются путем разделения тех данных, которые мы хотим сохранить, на отдельные фрагменты. Далее такие фрагменты размещаются в корзинах хранения данных, после чего происходит их хэширование и аналогичный процесс повторяется до тех пор, пока общее число хэшей не будет равно единице или корневому хэшу.

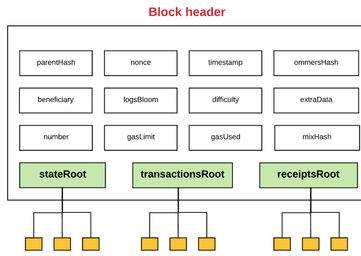


Для каждого значения, хранящегося внутри данного дерева, вам потребуется ввести определенный ключ. Для получения соответствующего значения, хранящегося в листовых узлах, вы должны получить команду ключа: цепочки какого дочернего узла необходимо придерживаться. Что касается Эфириума, то отображение ключа/значения, необходимого для дерева состояний, находится между адресами и связанными с ними учетными записями, в том числе balance, nonce, codeHash, а также storageRoot для каждой из учетных записей, при этом storageRoot является деревом.



Подобная структура префиксного дерева также может применяться для хранения как транзакций, так и страницы приема оплаты. Если останавливаться на этом более подробно, то каждый блок имеет так называемый «header» или заголовочный файл, в котором хранится хэш корневого узла трех разных структур дерева Меркла, в том числе:

- Состояние префиксного дерева
- Транзакции префиксного дерева
- Страницы приема оплаты для префиксного дерева

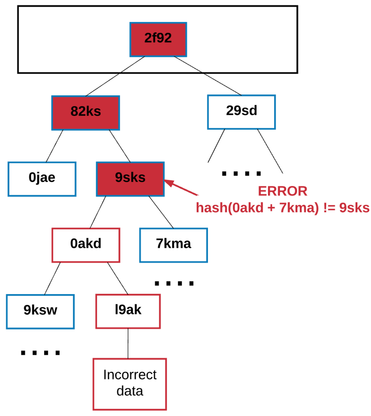


Возможность эффективного хранения данной информации в префиксном дереве Меркла Эфириума является невероятно практичным решением для так называемых тонких клиентов или тонких узлов. Необходимо также отметить, что поддержка блокчейна осуществляется с помощью набора узлов. Простыми словами: всего существует два вида узлов: полный и тонкий.

Полный узел архива выполняет синхронизацию блокчейна с помощью загрузки всей цепочки от блока первоначального состояния до текущего блока, содержащего заголовочный файл, при этом происходит выполнение всех располагающихся в нем транзакций. Как правило, майнеры хранят полный узел архива, поскольку без последнего у них не будет возможности участвовать в процессе майнинга. Кроме того, вы также может загрузить полный узел, при этом в выполнении каждой отдельной транзакции нет необходимости. Также стоит отметить, что каждый полный узел всегда содержит полную цепочку.

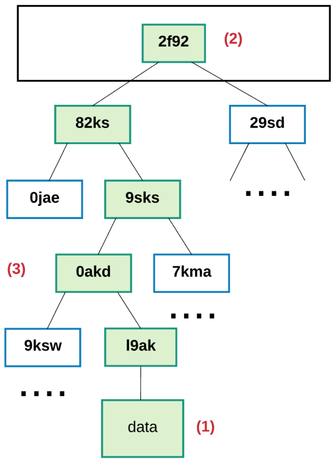
В том случае если для узла нет необходимости выполнять каждую отдельную транзакцию или запрашивать накопленные данные, то хранение полной цепочки может оказаться излишним. Именно в данном случае мы сталкиваемся с таким понятием как тонкий узел. **Вместо загрузки и хранения полной цепочки, а также выполнения всех транзакций, тонкие узлы загружают только цепочку заголовочных файлов из блока первоначального состояния в текущий заголовок, при этом не происходит выполнения каких-либо транзакций**. Поскольку у тонких узлов имеется доступ к заголовкам блоков, содержащих хэш трех префиксных деревьев, они могут с легкостью создавать и получать соответствующие ответы, касающиеся транзакций, событий, баланса и т.д.

Хэш в дереве Меркла распространяется от нижних ветвей к верхним, и если злоумышленник попытается заменить оригинальную транзакцию на поддельную в нижней части дерева Меркла, то это приведет к изменению хэша верхнего узла, а это, в свою очередь, приведет к изменению хэша располагающегося над ним узла и так до тех пор, пока, в конечном итоге, это не приведет к изменению корня.



Любой узел, для которого требуется проверка какой-либо части данных, использует так называемое «доказательством Меркла». Последнее состоит из:

- Фрагмента данных, который должен быть проверен
- Корневого хэша дерева
- Так называемой «ветви» – всех хэшей, от проверяемого фрагмента данных до корня.



Каждый пользователь, считывающий такое доказательство, может проверить, является ли хеширование для определенной ветви соответствующим на всем участке дерева, а также занимает ли данный фрагмент соответствующее положение в этом дереве.

Таким образом, можно сделать вывод, что преимущество применения префиксного дерева Меркла заключается в том, что корневой узел данной структуры является криптографически зависимым от данных, хранящихся в дереве. Следовательно, хэш корневого узла может быть использован в качестве безопасного идентификатора для этих данных. Ввиду того, что в заголовок блока включен корневой хэш деревьев, а также их состояния, транзакции и информацию о приходе оплаты, любой из узлов может проверять ту или иную часть состояния Эфириума без необходимости хранения всех состояний, которые могут быть потенциально неограниченным по размеру.

Горючее и вознаграждение

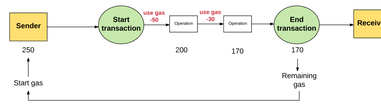
Одним из важных моментов в системе Эфириума является процесс оплаты. **За любое вычисление, осуществляющееся в результате проведения операций с транзакциями внутри сети Эфириума, берется определенная плата**. Номинал данной оплаты носит название «горючее» (от англ. gas).

Горючее – это единица измерения, которая используется для определения размера оплаты по конкретному вычислению. Цена на горючее – это количество «эфира», которое вы способны потратить на каждую единицу горючего. Измеряется цена на горючее в «gwei». Wei является самой маленькой единицей эфира, где 1018 Wei –это всего 1 эфир. Один gwei равен 1 000 000 000 Wei.

Для проведения любой транзакции отправитель должен установить **лимит горючего**, а также **цену на горючее**. Цена на горючее и лимит горючего – это максимальная сумма в Wei, которую отправитель готов заплатить за проведение транзакции.

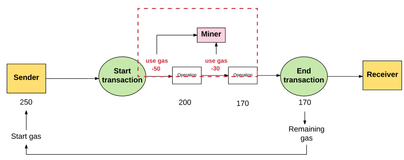
Давайте представим, что отправитель устанавливает лимит горючего в 50 000 gwei, а цену на горючее – в 20 gwei. Это значит, что отправитель готов потратить не более 50 000 x 20 gwei = 1 000 000 000 000 000 Wei или 0,001 эфира для проведения данной транзакции.

Таким образом, лимит горючего – это максимальное количество горючего, которое отправитель готов оплатить. В том случае если на балансе его счета достаточно эфира для покрытия данного максимума, то отправитель может проводить транзакции. Кроме того, отправителю возмещаются любые убытки, связанные с неполным использованием горючего по завершении транзакции, при этом горючее будет обменяно по первоначальной ставке.



В том случае, если отправителем не было предоставлено необходимого количества горючего для проведения транзакции, последняя будет проведена «без горючего» и будет считаться недействительной. Таким образом, проведение транзакции прерывается, а любые изменения состояния аннулируются, вследствие чего система Эфириум возвращает участников сделки в первоначальное состояние. Стоит отметить, что информация о такой неудавшейся транзакции записывается в системе, так что вы можете отследить, проведение каких транзакций было проведено и на каком этапе произошел сбой. И что также немаловажно: поскольку до того момента, как у отправителя закончилось горючее, машиной уже были затрачены определенные усилия для выполнения расчетов, логично было бы предположить, что убытки, связанные с расходованием горючего, уже **не будут возмещены отправителю**.

«Куда именно я отправляю горючее?» — спросите вы. Итак, **все деньги, которые были потрачены на приобретение отправителем горючего, отправляются на адрес бенефициария, который в большинстве случаев является адресом майнера**. Поскольку майнеры выполняют расчеты и проверку транзакций, то именно они получают плату за горючее в качестве вознаграждения.



Как правило, чем выше стоимость горючего, которую отправитель желает оплатить, тем выше оплата, получаемая майнером в результате проведения транзакции и, кроме того, тем более вероятно, что майнер сделает свой выбор в ее пользу. Таким образом, майнеры свободны выбирать, валидацию каких транзакции они желают осуществить, а какие транзакции им стоит проигнорировать. Зачастую, майнеры сообщают отправителям, какую цену им стоит задать для горючего, для того чтобы первые были готовы выполнять транзакции.

Оплата за использование хранилища

Горючее используется не только для оплаты тех или иных вычислений, но также и для оплаты за использование хранилища. Общая плата за использование хранилища составляет 32 используемых байта.

Вопрос оплаты, взымаемой за использование хранилища, имеет некоторые нюансы. Например, поскольку увеличение используемого в хранилище места подразумевает увеличение размеров базы данных состояний Эфириума, и это относится ко всем узлам, то у вас появляется стимул хранить только относительно небольшой объем данных. Таким образом, если какой-либо из этапов транзакции подразумевает удаление записи в хранилище, то оплата за выполнение этой операции не взымается, при этом ввиду освобождения места в хранилище убытки также будут возмещены.

Для чего предусмотрена оплата?

Важным аспектом работы Эфириума является то, что **любая операция, которая выполняется сетью, также одновременно выполняется каждым полным узлом**. Тем не менее, все шаги, связанные с вычислением на виртуальной машине Эфириума, очень дорого стоят. Таким образом, для решения простых задач (например, запуска простой бизнес-логики, проверки подписей, а также иных операций, связанных с криптовалютой) могут вполне сгодиться смарт-контракты Эфириума, в отличие от тех случаев, когда требуется выполнение других, более сложных, задач: хранение файлов или электронной почты, а также выполнение задач из области машинного обучения, которые могут вызывать чрезмерную загрузку сети. **Введение оплаты предотвращает действия пользователей, направленные на излишнюю загрузку сети.**

В Эфириуме используется полный по Тьюрингу язык. Если вкратце: машина Тьюринга – это машина, имитирующая любой компьютерный алгоритм. Для тех, кто впервые слышит о машине Тьюринга, предлагаю прочитать эту и вот эту статьи. Благодаря такой особенности в Эфириуме появляется возможность использовать циклы, и это делает его восприимчивым к проблеме остановки – проблеме, в случае возникновения которой, вы не можете определить, будет ли программа функционировать бесконечно или нет. Например, в том случае если бы в Эфириуме не была бы предусмотрена система оплаты, то злоумышленники могли бы попытаться сорвать работу сети, выполнив бесконечный цикл внутри транзакции, при этом не понеся каких-либо убытков. Таким образом, система оплаты была введена именно для ее защиты от преднамеренных атак.

Вполне вероятно, что вы подумаете: «А я-то тут причем? Зачем я буду платить за использование хранилища?» Ну, что вам сказать, вся сеть Эфириума берет на себя плату как за вычисления, так и за использования хранилища…как-то так.

Транзакции и сообщения

Ранее я уже писал о том, что Эфириум – **это система состояния транзакций**. Другими словами, благодаря транзакциям, которые происходят между различными учетными записями, происходит изменение или перемещение глобального состояния Эфириума из одного состояния в другое.

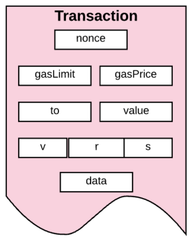
Проще говоря, транзакция является криптографически подписанной частью инструкции, которая сначала задается внешней учетной записью, а затем упорядочивается и передается в

блокчейн.

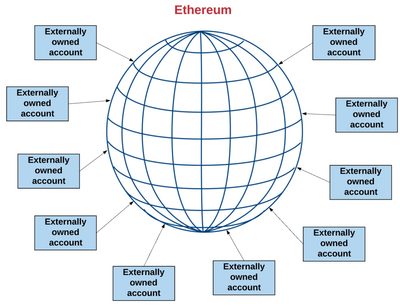
Всего существует два типа транзакций: **отправка сообщений** и **создание контракта** (другими словами, такие транзакции создают новые контракты в сети Эфириума).

Все транзакции содержат следующие элементы, вне независимости от типа первых:

- **nonce** – количество транзакций, которые были отправлены отправителем.
- **gasPrice** – количество Wei, которое отправитель готов отдать за единицу горючего, необходимую для совершения сделки.
- **gasLimit** – максимальное количество горючего, которое отправитель готов заплатить за проведение данной транзакции. Такая сумма задается и оплачивается заранее, прежде чем какие-либо вычисления будут проведены.
- **to** – адрес получателя. На момент выполнения транзакции, связанной с созданием контракта, адреса учетной записи контракта еще не существует, поэтому вместо него используется пустое значение.
- **value** – количество Wei, которые будут переданы от отправителя к получателю. В транзакциях, связанных с созданием контрактов, данная величина является стартовым балансом для вновь созданной учетной записи.
- **v, r, s** – данные обозначения, используемые для создания подписи, которая идентифицирует отправителя транзакции.
- **init** – предназначен только для транзакций, связанных с созданием контрактов. Фрагмент EVM-кода, используемый для инициализации вновь созданной учетной записи контракта. init запускается только единожды и в дальнейшем не используется. Когда init запускается в первый раз, данный элемент возвращает тело кода учетной записи, которое представляет собой часть кода, постоянно связанную с учетной записью контракта.
- **data** – это входные данные (параметры) для вызова сообщения (data является необязательным элементом, который предназначен только для вызовов сообщений). Например, если смарт-контракт играет роль службы регистрации домена, то вызов этого контракта может ожидать поля ввода (например, домен и IP-адрес).

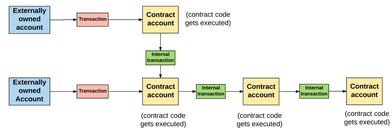


Из информации, приведенной в разделе «Учетные записи», мы выяснили, что транзакции – как для вызовов сообщений, так и для создания контрактов –инициируются внешними учетными записями, а затем перенаправляются в блокчейн. Другими словами, транзакции – это своеобразный мост, соединяющий внешний мир и внутреннее состояние платформы Эфириума.



Но это не значит, что одни контракты не могут взаимодействовать с другими: **контракты, находящиеся в глобальном контексте состояния Эфириума, могут взаимодействовать друг с другом в пределах данного контекста. Их взаимодействие или общение происходит с помощью отправки сообщений или внутренних транзакций**. Единственное отличие внутренних транзакций от обычных заключается в том, что первые не создаются внешними учетными записями – но в результате создания контрактов. Они являются виртуальными объектами, которые, в отличие от транзакций, не упорядочиваются и могут существовать только в среде исполнения Эфириума.

Когда один из контрактов осуществляет отправку внутренней транзакции другому контракту, выполняется определенный код, существующий в учетной записи контракта получателя.



Стоит также отметить, что для внутренних транзакций или сообщений **gasLimit** не предусмотрен, поскольку лимит горючего задает инициатор исходной транзакции (например, в какой-либо учетной записи). Лимит горючего, задаваемый внешней учетной записью, должен быть достаточно высоким для проведения транзакции, включая любые дополнительные действия, которые выполняются в результате проведения данной транзакции, например, передача сообщения от одного контракта к другому. В том случае если в цепочке транзакций и сообщений для выполнения одного из последних недостаточно горючего, то его выполнение, а также выполнение всех последующих сообщений, вызванных изначальным выполнением, будет возвращено.

Блоки

Все транзакции так или иначе сгруппированы в «блоки». Блокчейн содержит несколько таких блоков, соединенных между собой.

Такие блоки состоят из:

- заголовка блока
- информации о серии транзакций, включенных в данный блок
- серии других заголовков блоков для текущих оммеров

Что такое «оммеры»?

Давайте разберемся с тем, что такое оммер (от англ. «ommer»). Оммер – это блок, родителем которого является родительский элемент текущего блока. В этой главе я вкратце опишу, для чего вообще нужны оммеры, а также из каких соображений в блоке содержится заголовки блоков для оммеров.

Их наличие, в первую очередь, обосновано тем, что время блокировки в Эфириуме намного ниже (примерно 15 секунд), чем для других блокчейнов, например, для биткоинов (примерно 10 минут). Благодаря такой особенности скорость транзакций проведения увеличивается. С другой стороны, одной из негативных сторон более короткого времени блокировки является то, что борьба майнеров за очередное блочное решение только усиливается. Такие конкурирующие блоки еще называют «блоки без родителя» (т.е. такие блоки не входят в основную цепочку блоков).

Оммеры были созданы для того, чтобы майнеры могли получить заслуженную награду за включение блоков без родителей в основную цепочку. Оммеры, включенные майнерами в основную цепочку, должны быть «действительными»: они, оммеры, должны быть потомками в шестом или более раннем поколении текущего блока. Например, после шестого поколения такие потомки не могут быть включены в основную цепь в качестве блоков без родителя: более поздние транзакции могут негативно влиять на работу системы в целом.

За оммеры вы получите награду, меньшую чем за включение полного блока. Тем не менее, это не должно умалять попытки майнеров включить такие блоки без родителя и получить свою заслуженную награду.

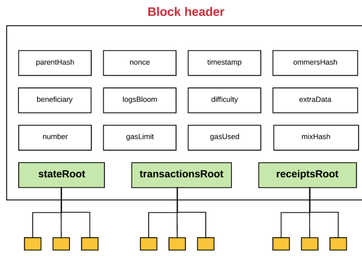
Заголовки блоков

Я уже упоминал ранее о том, что каждый блок имеет заголовок, но мы так толком и не разобрались, что это такое?

Заголовок блока – это часть блока, которая состоит из:

- **parentHash** – является хэшем заголовка родительского блока (благодаря чему, собственно, блок попадает в цепочку блоков)
- **ommersHash** – хэш текущего списка блоков оммеров
- **beneficiary** – адрес учетной записи, на который поступает оплата за включение этого блока
- **stateRoot** – хэш корневого узла состояния префиксного дерева (ранее я писал, что состояние префиксного дерева хранится в заголовке, тем самым для тонких клиентов упрощается процесс утверждения состояния)
- **transactionsRoot** – хэш корневого узла префиксного дерева, содержащий все транзакции, которые перечислены в данном блоке
- **receiptsRoot** – хэш корневого узла префиксного дерева, который содержит информацию об оплате для всех транзакций, перечисленных в данном блоке
- **logsBloom** – Фильтр Блума (структура данных), состоящий из информации, содержащейся в журналах
- **difficulty** – уровень сложности текущего блока
- **number** – номер текущего блока (первоначальный блок имеет номер, равный нулю; номер блока увеличивается на единицу для каждого последующего блока)
- **gasLimit** – текущий лимит горючего для текущего блока
- **gasUsed** – общее количество горючего, используемого для транзакций в текущем блоке

- **timestamp** – временная отметка, предназначенная для создания текущего блока
- **extraData** – дополнительные данные, которые касаются текущего блока
- **mixHash** – хэш, который в сочетании с элементом nonce утверждает, что для текущего блока выполняется достаточно вычислений
- **nonce** – хэш, который в сочетании с элементом mixHash утверждает, что для текущего блока выполняется достаточно вычислений



Стоит заметить, что каждый заголовок блока содержит три структуры префиксного дерева для:

- состояния (**stateRoot**)
- проведения транзакции (**transactionsRoot**)
- получения информации об оплате (**receiptsRoot**)

Такие структуры префиксного дерева являются ничем иным, кроме как префиксным деревом Меркла, которое мы уже рассматривали выше.

Кроме того, для такого определения существует несколько терминов, которые, вероятно, должны вам показаться интересными.

Журналы

В платформе Эфириума предусмотрена возможность вести журналы, цель которых записывать информацию о различных транзакциях и сообщениях. Кроме того, для контракта также существует возможность открытого создания записи в таком журнале с помощью объявления «события», которое требуется записать.

Запись журнала включает:

- адрес учетной записи регистратора
- ряд задач, которые отображают различные события, выполненные для текущей транзакции
- любые данные, которые имеют отношение к данным событиям

Записи журналов хранятся в Фильтре Блума, благодаря которому появляется возможность эффективно хранить бесконечное количество данных.

Получение информации об оплате

Записи, хранящиеся в заголовке, поступают из содержащейся в журнале информации, которая относится к данным об оплате транзакции (или чеку). Подобно тому, как вы получаете чек при покупке товаров в магазине, Эфириум создает подобный чек для каждой из транзакций. И как вы уже наверное догадались, в каждом чеке содержится информация о текущей транзакции. Чек включает в себя:

- номер блока
- хэш блока
- хэш транзакции
- количество горючего, используемого для текущей транзакции
- общее количество горючего, которое было использовано для проведения текущей транзакции, для определенного блока
- созданные по выполнении транзакции записи журнала
- другая информация

Сложность блока

Сложность блока – понятие, используемое для обеспечения согласованности времени, которое необходимо для валидации блоков. Для первоначального блока сложность составляет 131 072 единиц. Для вычисления сложности любого из блоков применяется специальная формула. В том случае если валидация одного из блоков произошла более быстро чем, например, валидация последующего, то протокол, используемый в Эфириуме, увеличивает сложность последнего.

Сложность блока также влияет на nonce – хэш, выполнение которого необходимо в течение отображения блока, при этом для данной цели применяются алгоритмы проверки безопасности.

Зависимость одного параметра, сложности блока, от другого, nonce, представлена в данной формуле:

$$n \leqslant \frac{2^{256}}{H_d}$$

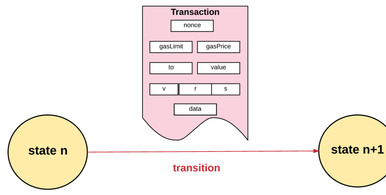
*где **Hd** — это сложность блока*

Единственный способ определить параметр nonce, который будет отвечать представленному формуле условию, заключается в использовании алгоритма проверки работоспособности для поиска всех его возможных значений. Ожидаемое время поиска всех соответствующих данному условию значений – это и есть сложность блока. Тогда мы можем сделать вывод: чем большее значение сложности блока, тем сложнее найти параметр nonce, и, таким образом, тем сложнее провести валидацию блока, в результате чего, в свою очередь, увеличивается время, необходимое для валидации последующих блоков. Значит, **на основании значения, полученного в ходе определения сложности блока, используемый протокол определяет, сколько необходимо времени для валидации текущего блока**.

В том случае если время, требуемое для валидации блока, оказывается меньше, чем ожидалось, то протокол занижает сложность текущего блока. Таким образом, время, требуемое для валидации блока, задается автоматически для постоянного соответствия текущим параметрам (в среднем такое время составляет 15 секунд).

Проведение транзакций

Ну, что ж, вот мы и подошли к, пожалуй, самой сложной части протоколов, используемых в Эфириуме – проведения транзакций. Давайте представим, что вы задали в сети Эфириума выполнение какой-либо транзакции. И что по вашему мнению, будет происходить с состоянием Эфириума, при проведении вашей транзакции?



Во-первых, любая транзакция должна отвечать определенным требованиям, для того чтобы ее выполнение не было отменено, а именно:

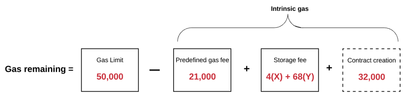
- Транзакции должны отвечать требованиям RLP. RLP – это рекурсивная длина префикса (от англ. Recursive Length Prefix), представляющий собой формат данных, который используется для кодирования вложенных массивов двоичных данных. Формат RLP используется в Эфириуме для упорядочивания объектов.
- Наличие валидной подписи транзакции.
- Наличие валидного значения nonce. Напомню, что nonce – это количество транзакций, отправленных с текущей учетной записи. Для того чтобы такое значение было валидным, оно должно соответствовать значению nonce для учетной записи отправителя.
- Лимит горючего для транзакции должен быть равным или больше заданного количества горючего. Заданное количество горючего включает:

1. предопределенная стоимость, равная 21 000 единиц горючего, необходимого для выполнения транзакции
2. Плата за горючее, используемое для отправки данных об транзакции (4 единицы горючего за каждый байт данных или код, равный нулю, и 68 – за каждый ненулевой байт данных или ненулевой код)
3. Дополнительные 32000 единицы горючего, если транзакция связана с заключением договора

- Баланс текущего счета отправителя должен содержать достаточное количество эфира для покрытия «авансовой» стоимости горючего, которую отправитель обязуется оплатить. Авансовая стоимость горючего вычисляется следующим образом: лимит стоимости горючего умножается на стоимость горючего для текущей транзакции, в результате чего мы находим максимальную стоимость горючего. Далее, к максимальной стоимости добавляется общее количество горючего, переправляемого от отправителя получателю.

В том случае если вы выполнили все указанные выше требования, вы переходите к следующему шагу.

В первую очередь, значение авансовой стоимости горючего вычитается со счета отправителя, а показатель nonce отправителя увеличивается на 1. После чего мы можем подсчитать оставшееся количество горючего по следующей формуле: **от общего количества горючего, необходимого для проведения транзакции, отнимаем заданное количество горючего**.



После этого начинается выполнение транзакции. В течение проведения текущей транзакции в Эфириуме происходит отслеживание «подсостояния». Подсостояние необходимо для записи информации, которая была собрана во время проведения текущей транзакции. Такая информация будет необходима сразу по завершении выполнения транзакции и содержит:

- **Self-destruct set**: набор учетных записей, которые будут удалены по завершении проведения транзакции
- **Log series**: заархивированные и проиндексированные контрольные точки, необходимые для выполнения кода виртуальной машины.
- **Refund balance**: сумму, которая должна быть возвращена отправителю по завершении транзакции. Я уже упоминал ранее, что использование хранилища, предусмотренного в Ethereum, стоит определенного количества денег, и эти деньги возвращаются отправителю после того, как он перестает использовать такое хранилище. В системе Ethereum сохраняется информация об использовании хранилища и возврата средств за его использование отправителю.

После этого происходит выполнение различных вычислений, необходимых для проведения транзакции.

После того как все шаги, необходимые для проведения транзакции, были завершены (при условии, что все указанные выше требования также были выполнены), состояние транзакции завершается, при этом происходит подсчет количества неиспользованного горючего, которое должно быть возвращено отправителю.

После завершения (удачного) транзакции и возврата горючего отправителю происходит следующее:

- определенное количество эфира, использованного для приобретения горючего, отправляется майнеру;
- горючее, использованное для проведения транзакции, записывается в блок для подсчета горючего (данный блок используется для хранения информации об общем количестве горючего, которое было использовано для проведения всех транзакций в данном блоке; кроме того, такой блок используется в течение валидации;
- вся информация об учетных записях, содержащаяся в разделе self-destruct set удаляется.

Вот мы и подошли к концу этой главы: узнали, что такое новое состояние и зачем для проведения транзакций нужен журнал.

В следующей главе мы более подробно ознакомимся с различием между транзакциями, связанными с созданием контрактов, и отправкой сообщений.

Создание контрактов

Вы наверняка помните, что в Эфириуме существует всего два типа учетных записей: контрактные учетные записи и внешние. Когда вы встречаете термин «транзакции, связанные с создание контракта», вы должны знать, что цель такой транзакции заключается в создании новой контрактной учетной записи.

Для создания новой контрактной учетной записи мы, в первую очередь, должны объявить адрес создаваемой учетной записи с помощью специальной формулы. После этого происходит создание новой учетной записи. Для выполнения такой операции вы должны провести ряд действий:

- выставить ноль в значении nonce
- настроить баланс вашей учетной записи, равный оплате за выполнение транзакции (в том случае если отправитель готов отправить некоторое количество эфира в качестве оплаты за проведение транзакции)
- подсчитываете размер оплаты, который переходит в баланс создаваемой учетной записи со счета отправителя
- указываете, что хранилище больше не используется вами
- настраиваете хэш-код контракта в качестве хэша пустой строки

В момент, когда мы приступили к созданию новой учетной записи, мы, по сути, уже ее создали с помощью **init-кода**, который автоматически отправляется в начале транзакции (забыли, что такое init-код – смотрите раздел «Транзакции и сообщения»). Существует несколько вариантов развития событий в течение выполнения init-кода (например, может произойти: обновление хранилища для учетной записи, создание другой учетной записи для текущего контракта, отправка сообщения и т.д.).

После выполнения системой кода, предназначенного для создания нового контракта, в игру вступает горючее. Вы не сможете провести транзакцию если для нее требуется большее количество горючего, чем то, которое хранится на вашем балансе. В том случае если несмотря на такое ограничение вы попытаетесь провести транзакцию, то вы получите сообщение об нехватке горючего, после чего система автоматически закроется. При этом если завершение транзакции было вызвано из-за нехватки горючего, то вас перебросит на стадию, предшествующую проведению транзакции. И что самое главное: **получателю не будет возмещено то количество горючего, которое было потрачено до обнаружения его нехватки**.

Вот такие дела…

Тем не менее, если отправитель выделил некоторое количество эфира для проведение текущей транзакции, то такое количество будет возвращено даже в случае неудачного завершения создания контракта.

Если инициализационный код был выполнен успешно, то средства, требуемые для создания контракта, должны быть внесены создателем. В данную сумму входит также стоимость использования хранилища, которая прямо пропорционально увеличивается с увеличением размеров созданного для контракта кода. В том случае если у создателя не хватает средств для проведения данной операции, то транзакция прекращается ввиду нехватки горючего и последствия будут теми же, что и приведенные выше.

Если же все прошло гладко, и мы не получили сообщения о нехватке горючего, то все неиспользованное для проведения данной транзакции горючее возвращается отправителю.

Победа!

Сообщения

Выполнение операции по отправке сообщения, в общем-то, достаточно схоже с созданием контракта, не учитывая некоторых небольших отличий.

Для выполнение данной операции применение init-кода совершенно не требуется, поскольку в результате ее выполнения не создается новой учетной записи. Тем не менее, для такой операции возможно понадобятся входные данные, но только в том случае, если такие данные были переданы получателем в результате проведения транзакции. После выполнения операции по отправке сообщения становится доступным новый блок, содержащий выходную информацию, использование которой происходит при повторном выполнении такой операции.

Также как и в случае с созданием контракта, если выполнение операции по отправке сообщения было прервано в результате нехватки горючего или недействительной транзакции (например, ввиду ошибки переполнения стека, неверного адреса перехода, неправильной команды), то количество использованного для данной операции горючего не возвращается инициатору вызова. Наоборот, все неиспользованное горючее также списывается с его баланса, и состояние системы возвращается в точку, предшествующую операции перевода баланса.

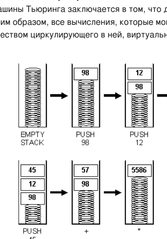
До недавнего времени в Эфириуме не было возможности прервать или приостановить выполнение транзакций без потери горючего, предоставленного вами для такой цели. Например, можно представить ситуацию, когда вы являетесь инициатором создания контракта, при создании которого произошла ошибка, поскольку инициатор звонка не имел права выполнять какую-либо из транзакций. Так вот, в предыдущей версии Эфириума, до обновления платформы, в такой ситуации все оставшееся на вашем счете горючее было бы снято, при этом отправитель также не получил бы обратно своего горючего. Но с **выходом обновления – Byzantium – у вас появляется возможность приостановить выполнение операций по созданию контрактов и вернуть систему в первоначальное состояние без потери оставшегося на вашем счете горючего**. Таким образом, если выход из транзакции произошел в результате приостановления ее выполнения, то неиспользованное горючее возвращается обратно отправителю.

Модель выполнения

В предыдущий разделах я рассказал вам о том, как происходит выполнение транзакций. Теперь же я предлагаю вам разобраться с тем, что происходит в VM (от англ. Virtual Machine — виртуальная машина) в момент выполнения транзакций.

Часть протокола, который выполняет обработку транзакций в операционной системе Эфириума называется виртуальной машиной Эфириума (ВМЭ).

ВМЭ является машиной Тьюринга, как это уже упоминалось в данной статье ранее. Единственное отличие ВМЭ от типичной машины Тьюринга заключается в том, что для работы первой требуется виртуальное «горючее». Таким образом, все вычисления, которые могут быть выполнены в ВМЭ, так или иначе ограничены количеством циркулирующего в ней, виртуальной машине, «горючего».



Источник: CMU

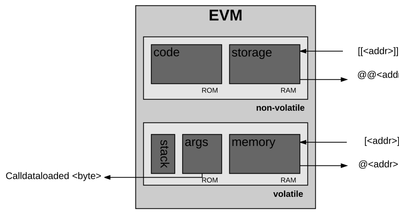
Кроме того, ВМЭ присущи все особенности стековой архитектуры. Стековая машина – это компьютер, в котором применяется алгоритм LIFO.

Размер любого элемента стека в ВМЭ равен 256 битам, а максимальный размер стека достигает 1024 битов.

Для ВМЭ предусмотрен некоторый объем памяти, который не является постоянным. В нем элементы хранятся в виде массивов байтов с обращением к словам.

Для ВМЭ также предусмотрено определенная область хранения. В отличие от объема памяти, такое

хранилище (или область хранения) не изменяется и является частью состояния системы. В ВМЭ программный код хранится в отдельной виртуальной ROM, доступ в которую можно получить только с помощью определенных инструкций. С этой точки зрения такая ВМЭ отличается от типичной архитектуры фон Неймана, в которой программный код хранится в памяти компьютера.



Для ВМЭ также предусмотрен свой специальный язык – байт-код ВМЭ. Когда программист, такой как вы или, например, я, пишет смарт-контракт, который будет выполняться в системе Эфириума, это обычно происходит с помощью высокоуровневого языка, такого как Solidity. После написания такого кода мы компилируем его в байт-код ВМЭ, чтобы ВМЭ могла понять написанную нами команду.

Перейдем непосредственно в выполнения операций.

Перед тем, как выполнить определенное вычисление процессор должен убедиться в том, что приведенная ниже информация является валидной и доступной:

- Состояние системы
- Информация о достаточном для выполнения требуемой операции количестве горючего
- Адрес учетной записи, которому принадлежит выполняемый код
- Адрес отправителя транзакции – инициатора выполнения текущей операции
- Адрес учетной записи – инициатора выполняемого кода (может отличаться от адреса отправителя-инициатора)
- Информация о требуемом для выполнения транзакции количества горючего
- Входные данные для выполнения операции
- Количество Wei, которое должно быть отправлено на счет данной учетной записи в результате проведения текущей операции
- Информация о выполняемом машинном коде
- Информация о заголовке блока для текущего блока
- Глубина выполнения текущего сообщения или создания контракта

Непосредственно до начала выполнения программы память системы является абсолютно пустой, а счетчик команд равен нулю.

```
PC: 0 STACK: [] MEM: [], STORAGE: {}
```

После чего в ВМЭ начинается рекурсивное выполнение транзакции: вычисление состояние системы и состояние машины для каждого цикла. Состояние системы – это глобальное состояние Эфириума. Состояние машины включает в себя:

- доступное количество горючего;
- счетчик команд;
- содержимое памяти;
- активное количество слов в памяти;
- контент стека.

Элементы стека добавляются или удаляются с левого края фрагмента кода.

Для каждого цикла из оставшегося количества горючего отнимается его определенная часть, при этом счетчик команд увеличивается.

Всего существует три возможных варианта окончания цикла:

1. Операции, выполняемые машиной, достигают исключительного состояния (например, ввиду нехватки виртуального горючего, неверных инструкций, недостаточного количества элементов стека, значения элемента стека, превышающего размер в 1024 бит, неверного назначения JUMP/JUMPI) и, таким образом, процесс выполнения операции приостанавливается.
2. Последовательность действий переходит к выполнению следующего цикла
3. Операции, выполняемые машиной, достигают логического завершения (завершения выполнения процесса)

В случае если вычисления, выполняемые машиной, достигают логического завершения, а не исключительного состояния, то в результате этого машина выдает результирующее состояние, а также информацию об оставшимся горючем и результирующие выходные данные.

Вот такие дела. Только что мы с вами усвоили самую сложную и запутанную часть Эфириума. Не переживайте если вы чего-то до конца не поняли: вам не нужно вникать в каждую мелочь и понимать все процессы, происходящие в данной системе, ну, если только вы не собираетесь по-настоящему полностью ее изучить и работать на достаточно глубоком уровне.

Окончательное оформление блоков

Давайте же наконец-то разберемся, что происходит с блоками транзакций во время их окончательного оформления.

«Окончательное оформление» может происходить по двум вариантам, в зависимости от того, создаем ли мы блок или он уже создан. В том случае если мы только создаем блок, то окончательное оформление означает процесс майнинга текущего блока. С другой стороны, если блок уже создан, то такое определение означает процесс валидации текущего блока. В обоих из представленных выше случаев необходимо выполнить четыре условия для окончательного оформления блока.

1) Валидация (или, в случае майнинга – определение) оммеров: каждый блок оммеров, который находится в заголовке блока, должен иметь валидный заголовок блока и быть шестым потомком текущего блока.

2) Валидация транзакций: значение gasUsed для текущего блока должно быть равным значению общего количества горючего, использованного для проведения всех перечисленных в данном блоке транзакций.

3) Назначение оплаты (только в случае майнинга): В адрес бенефициария назначается 5 единиц эфира за майнинг каждого блока (в соответствии с предложением EIP-649 данная оплата будет уменьшена до 3 единиц эфира). Более того, за каждый оммер, бенефициарию текущего блока назначается оплата в виде дополнительных 1/32 от общей оплаты за текущий блок. И последнее: бенефициарию блока оммеров также назначается оплата в виде определенной суммы, для определение которой существует специальная цифра.

4) Верификация состояния и значения nonce: Для проведения данной процедуры вам необходимо обеспечить выполнение всех транзакций, а также изменение результирующих состояний. После чего вам также потребуется задать новый блок после того, как оплата за данный блок была отправлена. Процесс верификации происходит посредством сравнения завершающего состояния с состоянием префиксного дерева, хранящегося в заголовке.

Майнинг, направленный на доказательство работы

В разделе «Блоки» мы кратко ознакомились с таким понятием, как сложность блоков. Алгоритм, благодаря которому возникло понятие сложность блоков, называется доказательство выполнения работы (PoW от англ. Proof of Work).

Алгоритм PoW, используемый в системе Эфириум, носит название Ethash (ранее, но назывался Dagger-Hashimoto).

Данный алгоритм имеет следующий вид:

$$(m, n) = \texttt{PoW}(H_{\not\mathbf{n}}, H_n, \mathbf{d})$$

где m – это mixHash; n – nonce; Hn – заголовок нового блока (nonce и mixHash не входят сюда, поскольку данные значения должны высчитываться); Hn – nonce для заголовка блока; d – комплект данных DAG.

В разделе «Блоки» мы также ознакомились с различными значениями, предусмотренными для заголовка блока. К ним, как вы помните, относятся такие значения как mixHash и nonce. Еще раз напомню:

- **mixHash** представляет собой хэш, который вместе со значением nonce подтверждает, что для текущего блока было выполнено достаточное количество вычислений.
- **nonce** также представляет собой хэш, который вместе со значением mixHash подтверждает, что для текущего блока было выполнено достаточное количество вычислений.

Таким образом PoW необходимо для подсчета вышеприведенных значений.

Объяснить каким именно образом mixHash и nonce высчитываются с помощью функции PoW достаточно сложная задача, и, собственно, этому моменту можно посвятить целую статью. Но если вкратце, то происходит следующее:

Значение «семени» рассчитывается для каждого из блоков. Для подсчета каждого из семени существует свой собственный «интервал», при этом каждый из интервалов равен 30000 блоков. Для каждого интервала семя является хэшем, равным серии из 32-байтовых нулей. Для каждого последующего интервала предусмотрен определенный хэш – хэш для предыдущего семени. С помощью данного семени узел находит значение псевдослучайного «хэша».

Такой хэш играет очень важную роль, поскольку с его помощью мы можем лучше понять, что собой представляют «тонкие узлы», речь о которых шла в предыдущих статьях. Целью тонких узлов является предоставление возможности для некоторых из узлов эффективно проверять те или иные транзакции без необходимости хранения всего набора данных блокчейна. Тонкий узел может провести валидацию транзакции, с помощью только данного хэша. Это происходит благодаря тому, что данный хэш может заново создавать необходимый ему для верификации блок.

Используя данный хэш, узел может создавать пакет данных DAG, в котором каждый элемент зависит от небольшого количества рандомизированных псевдо-элементов хэша. Каждый начинающий майнер должен для начала создать свой полный пакет данных. В системе для каждого из майнеров хранится отдельный пакет данных, при этом объем таких данных непрерывно растет.

Например, майнер может взять любые случайные части из пакета данных и использовать их в математической функции для того, чтобы хэшировать такие части для mixHash. Такой майнер сможет постоянно задавать значение для mixHash вплоть до получения исходных данных в виде значения nonce. Когда данное условие будет выполнено, то такое значение nonce будет считаться валидным, а блок может быть добавлен к цепи.

Майнинг как механизм защиты

В общем, целью PoW является криптографически доказать, что определенные вычисления были направлены на получение определенного результата (значения nonce). Так уж вышло, что **не существует другого способа нахождения nonce, значение которого не превышает определенного лимита, кроме как с помощь перечисления всех возможных вариантов вплоть до нахождения требуемого**. Распределение выходных данных для постоянно используемого хэша функций происходит равномерно. Таким образом, мы точно знаем, что **необходимое для нахождения значения nonce время явно зависит от порога сложности**: чем выше порог сложности, тем дольше будет происходить поиск необходимого значения nonce. Алгоритм PoW представляет концепцию сложности, используемой в рассматриваемом блокчейне.

Что же значит безопасный блокчейн? Ответ довольно прост: безопасный блокчейн – это блокчейн, которому будут доверять абсолютно ВСЕ ПОЛЬЗОВАТЕЛИ. Как я уже писал выше, в случае если в блокчейне существует более двух цепочек, то, вполне логично предположить, пользователи не будут чувствовать себя уверено во время работы с блокчейном, поскольку никто не сможет с точностью сказать, какая из представленных цепочек является валидной.

Именно **для этого применяется алгоритм PoW: обеспечивает единство цепочки в блокчейне, препятствуя созданию других цепочек блоков, которые могут повлиять на историю выполнения транзакций** (например создание ненастоящих транзакций или удаление или изменение существующих). Таким образом, для того чтобы злоумышленник смог первым провести валидацию своих блоков, ему придется постоянно определять значение nonce, при чем делать это быстрее всех других пользователей сети (надеюсь, вы помните про протокол GHOST, который я описывал ранее). Само собой, для злоумышленника такой метод будет неосуществим, если только в его распоряжении не находится большая часть майнинговых ресурсов сети – такой сценарий известен как атака 51 %.

Майнинг как средство распределения финансов

Помимо того, что алгоритм PoW обеспечивает безопасную работу блокчейна, благодаря ему также осуществляется распределение вознаграждения тем пользователям, чьи вычисления были использованы для обеспечения безопасности. Я уже писал выше, что майнеры получают вознаграждение за майнинг того или иного блока, а также:

- вознаграждение в 5 единиц эфира за «выигравший» блок (в скором времени эта цифра должна опуститься до 3 единиц)
- стоимость за горючее, израсходованное в результате проведения транзакции в блоке
- дополнительное вознаграждение за включение оммеров в блок

Для обеспечения согласованности работы метода PoW – что необходимо для гарантии безопасности – и распределения вознаграждений Эфириума постоянно придерживается двух принципов, представленных ниже:

- Во-первых, привлечь к использованию платформы как можно больше пользователей. Другими словами, использование данной платформы не должно вызывать у пользователя каких-либо сложностей: он не должен применять какие-то сверхсложные алгоритмы или задействовать неизвестное ему аппаратное оборудование. Кроме того, процесс распределения вознаграждения должен быть также ясен и прост для каждого, кто готов затратить некоторую энергию, используемую его компьютером, ради получения нескольких заветных единиц эфира.
- Во-вторых, не допускать несоразмерного распределения вознаграждений и других ресурсов для какого-либо отдельного узла: любой такой узел, для которого осуществляется несоразмерное распределение ресурсов, будет иметь огромное влияние на определение каноничного блокчейна, что негативно сказывается на безопасности системы в целом.

Например, в системе Bitcoin существует проблема с выполнением двух вышеприведенных принципов: ее алгоритм PoW использует хэш-функцию SHA256. Проблема последней заключается в том, что ее решение может оказаться намного проще в случае использование специальных аппаратных средств – ASICs.

Для того чтобы не допустить подобных проколов в Эфириуме применяется специальный алгоритм PoW с последовательной памятью (Ethhash). структура алгоритма построена таким образом, что для вычисления значения nonce требуется задействовать большой объем памяти и высокую пропускную способности соединения. Требования, связанные с наличие большого объема памяти, подразумевают, что для компьютера с обычным объемом памяти будет очень сложно провести параллельное вычисление нескольких значений nonce одновременно. Что до требований к высокой пропускной способности, то даже для сверхбыстрого компьютера обнаружение нескольких значений nonce одновременно станет непростой задачей. Таким образом, благодаря таким особенностям данной системы обеспечивается снижение вероятности централизации рисков и, кроме того, создаются более равномерные условия для работы различных узлов, выполняющих верификацию.

Кстати, не так давно я узнал о том, что Эфириуме собирается переключиться с алгоритма PoW к некоему методу, который носит название «Доказательство доли владения» (от англ. Proof-of-stake). Такой метод сам по себе достоин отдельной статьи для обзора и обсуждений.

Заключение

Ну, вот мы и подошли к логическому завершению нашей статьи.

На самом деле данная статья дает много пищи для размышлений. Вам совершенно не стоит переживать на тот счет, если вы осилили данную статью со второго или третьего раза. Я лично перечитывал yellow paper и white paper для Эфириума множество раз, прежде чем начал вникать в суть дела.

Я очень надеюсь, что данная статья все-таки оказалась для вас полезной. Если вы нашли какие-то ошибки, я буду вам очень благодарен, если вы сообщите мне о них.

Источники:

github.com/ethereum/yellowpaper
medium.com/@preethikasireddy/how-does-ethereum-work-anyway-22d1df506369

Метки: ethereum, эфириум, криптовалюта

+33 196 25,9k 21

Подписчики
8,0 Карма 16,5 Рейтинг 3
Денис Зарицкий @zarytskiy
Пользователь

Комментарии 21

Costic 21.10.17 в 19:06 +1

Такой статьи не хватало. Жаль только примера использования нет. На C++ или на JavaScript что-нибудь вроде HelloWorld.

vikarti 22.10.17 в 05:08 +3

1018 Wei = 1 Eth? а точно не 10^18? (в комментах автору оригинальной статьи на это указали несколько раз).

Примеры использования контрактов кстати есть на хабре (на Solidity — JavaScript-подобный язык), например habrahabr.ru/post/312008 и habrahabr.ru/post/327236

Ну и в дополнение про дыры — habrahabr.ru/company/neobit/blog/324456

aleks_raiden 22.10.17 в 14:19 +1

Большое спасибо за отличную статью!

ftdgoodluck 22.10.17 в 15:12 +1

Годная статья, спасибо!
Теперь есть куда отправлять друзей вместо попыток самому донести информацию

SbWereWolf 22.10.17 в 20:32 −1

прочитал минимум две трети статьи и как то понимания не прибавилось.
«время блокировки в Эфириуме намного ниже (примерно 15 секунд), чем для других блокчейнов, например, для биткоинов (примерно 10 секунд)» такие вещи вообще сводят с ума. примерно 15 секунд намного ниже чем примерно 10 секунд?
или перевод отвратительный или статья отвратительная.

zarytskiy 22.10.17 в 21:38 +1

Не стоит делать вывод на основании опечатки, во втором случае 10 минут, а не секунд. По поводу понимания: «На самом деле данная статья дает много пищи для размышлений. Вам совершенно не стоит переживать на тот счет, если вы осилили данную статью со второго или третьего раза. Я лично перечитывал yellow paper и white paper для Эфириума множество раз, прежде чем начал вникать в суть дела.»

debounce 22.10.17 в 21:34 +1

Вкуснота какая! Спасибо

Как контракт подписывает свои транзакции? Внешние аккаунты имеют закрытый ключ, поэтому могут подписать транзакцию, а майнер и другие узлы проверить подпись. А как подписать не имея ключа? Какой механизм используется, это можно описать простыми словами Алиса-Боб?

debounce 22.10.17 в 23:02 +1

Интересно про пропускную способность сети, стоимость вычислений и хранилища.
Мне попадались такие цифры
Вычисления: простая операция(сложение двух чисел, например) — $25 за миллион операций
Хранение в блокчейне: 1КБ — $5, 1МБ — $5 тыс, 1ГБ — $5 млн.

Примеры Solidity рекомендуют хранить данные в словарях, что-то типа mapping(address => uint) public balances;
Получается, уже одна такая запись address 160 бит + uint 256 бит = 416 бит = 52 байта. 20 таких записей в массиве = $5 за хранение, 1000 записей = $250
Плюсом ограничение на пропускную способность сети.
Как построить что-то массовое при таких лимитах? Банк, например, или платежную систему? Какие-то «внешние» решения?

Darth_Malok 23.10.17 в 05:44 +2

Забавно. Раньше мне эфириум представлялся биткоином 2.0, в котором просто внутренний язык скриптов сделали тьюринг-полным, но теперь я вижу, что всё намного сложнее.
Не совсем понял, зачем нужно было вводить сущность «топливо»? Почему не платить за каждый оператор виртуальной машины эфиром? И стоимость должна определяться динамически — майнерами. Выделил на контракт мало валюты — майнеры не заинтересовались, код контракта не выполняется. Всё по аналогии с комиссиями транзакций в биткоине. Сейчас же в эфириуме слишком много зависит от жёстко зашитых переменных в коде — система слишком централизована.

С хранением данных тоже как-то перемудрили. Неужели сложно было так же сделать некий «аукцион» —

ЧТО ОБСУЖДАЮТ

Сейчас Вчера Неделя

Skype For Linux перестал поддерживать процессоры AMD старше 5 лет
7,5k 50

Президент России провел заседание Совбеза, посвященное информационной безопасности
3,7k 28

Европейские компании обеспокоены запретом VPN в России
17,3k 55

Основная причина роста стоимости Bitcoin
12,7k 58

Почему у нас будут доказательства существования пришельцев – если они существуют – к 2035 году
32,5k 197

кто больше платит, те данные и храним?

Я понимаю, что я далеко не Виталик, и, наверное, по другому сделать было нельзя по тем или иным причинам, но блин… Хочется чего-нить такого — geektimes.ru/post/222493, чтобы восхищаться гениальностью эфиринума, как я восхищаюсь простотой и гениальностью биткоина, но пока эфириум мне нравится всё меньше и меньше.

> **worldmind** 24.10.17 в 17:09 +1
>
> да, тоже хотелось бы подробного объяснения причин введения топлива.

> **Closius** 23.10.17 в 19:15 0
>
> Транзакция считается корректной только тогда, когда она прошла процесс проверки – так называемый «майнинг»

Про какую транзакцию идет речь? А если я хочу передать некоторое кол-во эфиров с одного кошелька на другой кошелек — это же транзакция и для этого надо майнить? Лично мне не очень понятно.

> **i-vitalii** 26.10.17 в 20:31 0
>
> Да, это транзакция, которую надо майнить. Вообще весь блокчейн держится на нодах и майнерах. Нода — это приложение-база данных о всех подтвержденных транзакциях в системе и неком пуле новых неподтвержденных. Нода на связи еще с несколькими нодами, чтобы оперативно получать / отдавать изменения. Майнеры цепляются к ноде, выбирают из пула самые вкусные транзакции (где вознаграждение побольше) и формируют из них новый блок определенного размера. Далее начинают подбирать хеш определенного вида для этого блока. Если успевают вперед других — то нода всем рассылает обновления и все майнеры начинают работать со следующим блоком (возможны коллизии, они решаемы без проблем). Транзакция считается более-менее надежно подтвержденной, когда ее хотябы три майнера подтвердили — меньше шансов, что ее откатят по коллизии.

> **worldmind** 24.10.17 в 17:11 +1
>
> Что-то нигде не описывают механизм обмена сообщениями — как узел инициирующий транзакцию рассылает её всем остальным узлам? Как потом скачивается новый блок?
> Должен же быть какой-то протокол всех этих взаимодействий.

> **a5b** 25.10.17 в 02:45 +1
>
> Обнаруживаются узлы (для начала процесса нужно знать один адрес любого узла из сети) и скачиваются блоки видимо по Kademlia-подобной P2P сети (без DHT)
> https://github.com/ethereum/devp2p/blob/master/rlpx.md
>
> > The current version of RLPx provides a network layer for Ethereum.…
> > RLPx utilizes Kademlia-like routing which has been repurposed as a p2p neighbour discovery protocol.
> > Node discovery and network formation are implemented via a kademlia-like UDP.
>
> https://github.com/ethereum/wiki/wiki/Ethereum-Wire-Protocol
>
> > Two peers connect & say Hello and send their Status message. Status includes the Total Difficulty(TD) & hash of their best block.
> > The client with the worst TD asks peer for full chain of just block hashes.
> > Chain of hashes is stored in space shared by all peer connections, and used as a "work pool".
> > While there are hashes in the chain of hashes that we don't have in our chain:
> > Ask for N blocks from our peers using the hashes. Mark them as on their way so we don't get them from another peer.
>
> Сообщением NewBlock нашедший сообщает пирам о новом блоке. Кто предъявит более длинную цепочку, тот и прав (Consensus is based on choosing the block with the highest total difficulty. — https://github.com/ethereum/wiki/wiki/Mining). Пиры обмениваются новыми блоками (и видимо регулярно сообщают о новых total difficulty).
>
> Сообщением Transactions любой узел предлагает транзакцию для помещения в очередь.
> Эти сообщения узел рассылает своим 5 пирам, а каждый из них может переслать своим пирам.

> **worldmind** 25.10.17 в 09:54 0
>
> > Сообщением NewBlock нашедший сообщает пирам о новом блоке. Кто предъявит более длинную цепочку, тот и прав
> > Сообщением Transactions любой узел предлагает транзакцию для помещения в очередь. Эти сообщения узел рассылает своим 5 пирам, а каждый из них может переслать своим пирам
>
> вопрос масштабирования, допустим блок кто-то рассчитал — как быстро он распространится по всей сети? Аналогично и с транзакцией при условии что сеть охватила всю планету? Вроде в биткойне есть какие-то полные ноды, которые как бы центральные узлы и видимо должны ускорять процесс распространения информации.

> **a5b** 25.10.17 в 16:03 +1
>
> Полные ноды — не центральные узлы, а узлы p2p сети. В Эфирной сети тоже есть полные ноды (но вроде можно не сразу выкачивать весь блокчейн, а делать запросы getBlock когда потребуется). В https://ethereum.stackexchange.com/questions/4108/what-are-the-valuable-benefits-of-having-a-node пишут что без полной ноды для отправки транзакции нужен RPC-запрос к полной ноде (Light-client-protocol?). Полные ноды будут важны после (постоянно откладываемого) перехода с PoW (gpu-майнинг) на PoS (Форжинг/Минтинг).
>
> Транзакции распространяются по сети за секунды (до десятка) — https://ethereum.stackexchange.com/questions/5699/how-long-is-it-from-the-time-a-transaction-is-sent-until-it-is-viewable-on-pendi — за счет того что в p2p сети каждый узел знает еще к узлов (например 5; они выбираются по псевдослучайному алгоритму) и производят broadcast как новых транзакций, так и блоков (эфирные блоки небольшие, размером в килобайты) — т.е. рассылка как в биткоине — https://bitcoin.stackexchange.com/questions/32237/how-are-transactions-broadcast-to-the-rest-of-the-network, https://bitcoin.org/bitcoin.pdf 5. Network — 1) New transactions are broadcast to all nodes… 4) When a node finds a proof-of-work, it broadcasts the block to all nodes.
>
> В эфире есть вознаграждение не только за блоки в основной (самой длинной) цепочке, но и за свежие Uncle Blocks — https://bitcoin.stackexchange.com/questions/39329/in-ethereum-what-is-an-uncle-block "Uncles are stale blocks, ie with parent that are ancestors (max 6 blocks back) of the including block. Valid uncles are rewarded in order to neutralise the effect of network lag on the dispersion of mining rewards, thereby increasing security. Uncles included in a block formed by the successful PoW miner receive 7/8 of the static block reward = 4.375 ether A maximum of 2 uncles allowed per block."
>
> Не нашел размера сети в узлах, но на ethstats.net можно наблюдать за жизнью около сотни узлов (которые были добавлены на сайт) и за скоростью рассылки нового блока между ними. 75% блоков распространяются менее чем за 1,5 секунды, 90% блоков распространяются за 6 секунд. Там же видны какие-то peer count и pending transactions на узлах.

> **worldmind** 25.10.17 в 16:21 0
>
> Благодарю за подробные объяснения, хотя то что сейчас транзакции и блоки относительно быстро расходятся по сети не означает что также будет с ростом сети, и даже не означает что зависимость линейная.
> Кстати, мне кажется и Pow и PoS (с владением денег) это пустые идеи, никакой децентрализации тут не получится, как видно на примере биткойна мощности вполне можно сосредоточить в одних руках, а уж в возможности сосредоточить в одних руках деньги никто и не должен был бы сомневаться — мы живём в мире где 1% населения владеют почти половиной богатств.
> Чтобы была децентрализация узлом сети должен быть реальный человек. Один человек — один голос.

> **a5b** 25.10.17 в 16:47 +1
>
> > зависимость линейная.
>
> Зависимость в p2p сети (для диаметра сети и для количества "рукопожатий" от числа узлов) логарифмическая: https://en.wikipedia.org/wiki/Distributed_hash_table *"Beyond basic routing correctness, two important constraints on the topology are to guarantee that the maximum number of hops in any route (route length) is low, so that requests complete quickly; and that the maximum number of neighbors of any node (maximum node degree) is low, so that maintenance overhead is not excessive. Of course, having shorter routes requires higher maximum degree."*
>
> Проблемы масштабирования — в первую очередь из-за быстрого роста объема блокчейна (десятки ГБ? https://www.reddit.com/r/ethereum/comments/5om2tw/current_ethereum_blockchain_size/, https://twitter.com/TuurDemeester/status/874149251462434816/photo/1), а не из-за роста размера сети.
>
> Биткоин не в одних руках, а в 12 пулах, крупнейшие из которых имеют 17, 14, 12, 11 % — https://blockchain.info/pools от 340+ MB? https://geektimes.ru/post/245936/#comment_8364378 (~тонна CO2 за 1 биткоин, ~0.5 млн долларов в сутки на оплату электричества)

> **worldmind** 25.10.17 в 16:55 0
>
> Ну сеть пока очень мала, хотя логарифмическая зависимость это неплохое масштабирование.
> С объёмом блокчейна проблема не очень понятна — надо лишь залежавшиеся койны поднимать в свежие блоки (делать свежую транзакцию на тот же кошелёк), тогда в старых блоках ничего нужно не будет и их можно будет выбрасывать. Точнее тот кто в сети не участвует и не поднимает свои койны тому придётся платить за проведение транзакции нуждающейся в полном блокчейне.
> Да, помимо того что 4 пула уже больше половины сети составляют, они ещё и вхолостую энергию в тепло переводят, просто супер методика.

> **Darth_Malok** 28.10.17 в 06:06 0
>
> Во первых, пулы майнят не своими мощностями, а мощностями пользователей. Если какой-то пул начнёт играть не по правилам, и об этом станет известно сообществу, существенную часть своих мощностей такой пул потеряет, так что всё в общем-то неплохо на данный момент.
>
> Насчёт электроэнергии… А сколько энергии вхолостую в тепло переводит VISA или PayPal? А сколько человекочасов уходит на поддержание этих систем?) Поддержка криптовалют гораздо более экономична.

> **worldmind** 28.10.17 в 08:47 0
>
> А что значит мощности пользователя? Это раньше это был компьютер дома у человека, а сейчас это железяки в далёком китайском датацентре, они может формально и принадлежат каким-то пользователям, но при необходимости их легко изымут в пользу Китайского правительства или кого-то ещё.
>
> > Насчёт электроэнергии… А сколько энергии вхолостую в тепло переводит VISA или PayPal?
>
> сколько? и как она растёт со временем и с количеством транзакций?

Только полноправные пользователи могут оставлять комментарии. Войдите, пожалуйста.